Proj2: CUDA and Python Integration with applications using a project template

Deadline: 11/26/2024 (in TWO weeks, BEFOR Thanksgiving)

What to submit: submit your modified template programs through Blackboard

Description: Based on the techniques discussed in F24\_L8\_Integration.ppt, implement the haversine distance formula in CUDA, using the provided project template to enable calling CUDA/C++ APIs in Python code through shared library.

#1 The CUDA code you need to implement is the haversine\_distance\_kernel(…)global function in haversine\_library.cu. You can use any references for the Haversine distance formula. If you reuse an online implementation, please provide a reference (e.g., URL).

#2 Modify haversine\_distance(…) function in haversine\_libary.cpp, use the provided calc\_time(…) function to measure the runtimes of GPU memory allocation, CPU->GPU data transfer, running the kernel, and finally GPU->CPU data transfer.

#3 test\_3cities.py compute pairwise Haversine distances among three cities: "New York", "Paris", "Sydney" with (x,y) representing (longitude, latitude), respectively. Verify that output matches the expected using “assert np.allclose(…)”.

#3 test\_nycyellowcab.py uses monthly NYC yellow cab trip record data, which can be downloaded from <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>. The data is in the popular parquet format, which is supported by many packages such as Pandas and cuDF. The resulting dataframes allow you to perform further operations such as filtering and other types of queries. Learn APIs of both Pandas and cuDF on filtering/query (e.g., <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.query.html> and <https://docs.rapids.ai/api/cudf/stable/user_guide/api_docs/api/cudf.dataframe.query/>) . Your coding is not needed in this step.

#4 Download all the 12 months data in 2019, write a loop to read their parquet data, filter out records for trips whose pickup locations(Start\_Lon, Start\_Lat) or drop locations (End\_Lon, End\_Lat) that are not in NYC bounding box [**-**74.15, 40.5774, **-**73.7004, 40.9176]. Combine the monthly trip dataframes into a single large dataframe, convert the [Start\_Lon, Start\_Lat, End\_Lon, End\_Lat] columns to numpy arrays before calling haversine\_library.haversine\_distance(…) to compute the distances between pickup and drop-off pairs.

#5 Re-implement the Haversine distance function using Python, loop over all the filtered x1/y1/x2/y2 arrays in Step 4 and measure the runtime (using timeit package or alike). Compare your Python implementation runtime with the CUDA-accelerated runtimes, using both kernel runtime you have measured in Step 2 and the end-to-end runtime you have measured using timeit in Python code that calls the GPU accelerated library.

#5 (bonus) Create histograms for Start\_Lon, Start\_Lat, End\_Lon, End\_Lat and distance arrays, respectively and draw their histograms using Matplotlib APIs.

#6 (bonus) Continue with #3 and learn how to do grouping and aggregations in Pandas and/or cuDF. Compute numbers of trips and average distances of weekday/weekend trips.

**Have fun!**